A Case Study in Formal Performance Evaluation of an E-Voting System Using PEPA

Mohammed Alotaibi and Nigel Thomas*

Abstract

Performance overhead introduced by security properties of e-voting schemes needs to be investigated to have an insight on the average response times that voters will observe when they cast their votes using remote electronic voting systems. Timely responses of remote electronic voting protocols are important to increase voters' confidence in e-voting systems. In this paper we will study the individual verifiability impact of e-voting schemes on average response times of large scale e-voting scheme known as DRE-i by using the well-known formal stochastic performance evaluation process algebra language, PEPA. We will present a PEPA model of the e-voting scheme and show the response time analysis when voters verify the integrity of their votes.

Keywords: *Performance modelling, PEPA, remote electronic voting.*

1 Introduction

Today, we live in an unprecedented widespread usage of internet-connected digital devices that help services' consumers to access and consume internet-based electronic services in a relatively secure and reliable way. One influential and emerging service is the electronic voting where many countries have used this service electronically to increase the participation and turnout of voters in political elections [16]. Scalability and timely responses are two crucial implementation requirements for large scale e-voting systems and investigating the impact of security complexity in e-voting schemes on scalability and timely responses will help in designing more scalable and efficient e-voting systems. Recently, the lack of proper performance evaluation for the scalability and timely responses of the vote registration site of the UK government led to the crash of the site hours before the registration deadline due to a burst in traffic [22].

Most of the research done on the evaluation of e-voting schemes was related mainly to the evaluation of security properties of the e-voting systems such as proving security requirement correctness in [21], [14] and [1], and little research was done to investigate the performance aspects of e-voting schemes. In this paper, we are concerned with the formal evaluation of the response time of large scale DRE-i e-voting scheme [10]. Quantifying the response time of e-voting

*Newcastle University, School of Computing Science, Newcastle upon Tyne, UK, M.Alotaibi1, Nigel.Thomas@newcastle.ac.uk

systems will help e-voting systems' designers and security managers to build and maintain secure, dependable and cost-effective electronic voting environments. In particular, we are interested in the effect of individual verification actions carried out by voters on the average response time that voters expect when they cast their votes. The next section will provide the reader with a brief background about the basic concepts of e-voting schemes and the performance evaluation formalism using performance evaluation process algebra. In section three, we will describe the behaviour of the DRE-i e-voting scheme and construct and analyse the average time response of cast votes using PEPA. At the end, in section four, we will present our conclusions.

2 Background

In this section we will provide a concise background which is closely related to the formal performance analysis of secure electronic voting systems using stochastic performance evaluation process algebra. The first subsection will cover in brief the fundamental concepts of electronic voting schemes and the second subsection will cover the main concepts of stochastic performance evaluation process algebra.

2.1 Electronic voting

In general cryptographic voting schemes consist of the following entities: voters, election authorities, candidates and adversaries. On election day, eligible voters prove their identities to the election authorities to proceed and choose their preferred candidates on ballots and cast their ballots in ballot-boxes. Because voting process requires high level of privacy to protect the secrecy of the voters' choices and identities, the e-voting schemes have to satisfy some security requirements that can be delivered through well-known cryptographic building blocks.

2.1.1 E-voting security requirements

Electronic voting security literature identified many security requirements for e-voting protocols such as: completeness, privacy, soundness and robustness, receipt-freeness, verifiability, fairness, eligibility, and unreusability. The following is a brief description for the main e-voting security requirements and readers can refer to [9] for more details.
Completeness(correctness). The secret ballot protocol is considered complete (correct) if valid votes are counted correctly [7].
Privacy. The secret ballot protocol is considered privacy preserving if neither the voter nor any other participant can link the voter with the ballot that he cast. This property will prevent coercing and intimidating voters.
Soundness (robustness). In [7], the secret ballot protocol is considered sound if the invalid votes are not counted in the final vote tally. Another related requirement is the robustness of the secret ballot protocol which ensures that the voting protocol can deliver the required results if faulty operations do not exceed a defined threshold.
Receipt-freeness. A secret ballot protocol is considered receipt-free if no voter

can obtain or construct a proof that discloses how the voter voted. This property will prevent vote selling and coercion.
Verifiability. A secret ballot protocol is called verifiable if anyone of protocol participants can construct a proof that shows the protocol behaved correctly.
Fairness. A secret ballot protocol is called fair when no one can perform partial tally before publishing tally result except the voter who already knows his cast vote.
Eligibility. A secret ballot protocol has the eligibility property if only authorised voters are allowed to express their choices by casting their ballots. Election authorities identify and register voters who are eligible to vote and during the vote-casting day, the authorities perform identification and authentication services to prove the eligibility of each voter to vote. This property will prevent fraudulent votes.
Unresuability. A secret ballot protocol is called un-reusable if the voter is restricted to one valid vote. The last valid ballot cast by the voter will be considered his vote and will be used in vote tallying process.

2.1.2 E-voting Cryptographic Building Blocks

To achieve these security features, the electronic voting schemes use different cryptographic building blocks, which include blind signatures, mix-nets, encryption algorithms, and interactive and non-interactive proofs.
Encryption and signature primitives. Secret ballot protocols use encryption schemes to provide privacy for voters and use digital signature schemes to bind votes to voters in a non-deniable way.
Blind signature. It was proposed by Chaum [3] to hide the message's content to be signed from the signer and to certify the validity of the message by a trusted third party (the signer). Blind signature can be used in electronic voting to enable voters to get their votes certified by an election-trusted party as eligible to be counted.
Mix-nets. Mix-nets were proposed by Chaum [4] for anonymous communication which is based on cryptography and random permutations and they are used in e-voting to break the link between the voter and the vote's receiver. There are two types of Mix-nets protocols: decryption mix-nets [4] and re-encryption mix-nets. In the decryption mix-nets the protocol starts with the encryption phase where the voter's encrypted vote is encrypted sequentially with the public keys of the mix-servers and the encrypted vote will be passed to the first mix server. Then each mix-server will partially decrypt the encrypted vote received from the previous mix-server, shuffle the order of the decrypted votes, and pass them to the next mix-server. The last mix-server will publish the result on a bulletin board. In the re-encrypt mix-net, each mix server will re-encrypt the received encrypted vote, shuffle the votes, and pass votes to next mix servers. The last mix server will decrypt the encryption of the previous mix servers and post the result on a bulletin board.
Homomorphic encryption. To satisfy the privacy requirement of secret ballot protocol , the cryptographic based voting protocol may use some homomorphic encryption algorithms such as RSA [20], ElGamal [6], and Pailler [18]. It allows applying certain operations on sets of encrypted values (votes) without decrypting them. By using suitable homomorphic encryption primitives in electronic voting , tallies can be calculated from the encrypted ballots without

revealing the choice of the voter.

Interactive and non-interactive proofs. Some electronic voting schemes provide the voting scheme participants with the capability of verifying the validity of some voting activities through the interactive or non-interactive proofs. Interactive proof is a cryptographic protocol with two participants, the prover P and verifier V. Through the protocol interactions, the prover tries to prove the knowledge of a secret to the verifier and at the end of the protocol the verifier either accepts or rejects the proof constructed by the prover.

Based on the building blocks used to preserve vote privacy, we can identify three main cryptographic voting approaches: blind signature, homomorphic and mix-nets. In blind signature approach, one election authority gives the voter a token to confirm the eligibility of voter to vote. The token is a secret message known to the voter and signed by the election authority and will be submitted with the vote through anonymous channel to the tallying authority. In homomorphic-encryption approach, the voter's encrypted vote will stay encrypted during tallying process and encrypted votes will be counted using a property of homomorphic encryption. Finally in mix-net e-voting approach, the link between the vote and voter is hidden by using layers of encryption and random shuffles of batches of votes.

2.2 Performance Evaluation Process Algebra (PEPA)

Process algebra is an abstract language used for formal specification and design of concurrent systems. Process algebra languages are used to model collections of entities and their behaviour and the performance evaluation process algebra (PEPA) is a well-known process algebra language with stochastic extension [12].

Because PEPA models underlie Continuous Time Markovian Chain, some performance metrics can be derived from the steady state probability distribution of PEPA models, such as throughput, utilisation and population levels. However, due to state space explosion problem associated with large PEPA models when deriving their state space, Jane Holliston in [13] presents a new approximation approach using fluid analysis for performance analysis in PEPA models. Her approach is very useful when modelling systems with very large number of components because the fluid analysis approach does not depend on the derived state space of the PEPA model. The new approximation approach constructs a matrix called the activity matrix from the components and activities of the PEPA model, and by using the activity matrix the new approach can generate a set of ordinary differential equations that approximately represent the behaviour of the PEPA model. The system of ODEs can be solved and thus the performance metrics of the PEPA model can be calculated.

PEPA Syntax. PEPA is a markovian process algebra that models systems behaviour in terms of components and activities to evaluate their performance. Activities have actions that happen during exponentially distrusted intervals, and these activities when occur will change the components internal states. Therefore, the underlying state space of PEPA model can be used to construct Continuous Time Markov Chain (CTMC) and then the CTMC can be used to derive the steady state probability distribution of the model. PEPA uses a few combinators to construct formal stochastic process algebra models in a compositional way. The following represents a brief description for PEPA operators and readers can investigate [12] for more formal description of PEPA operators.

Prefix (.) : (α, r).P . The component can undertake an activity of type α at rate r and evolves into component P. The rate r is fixed during the lifecycle of the model and the passive (unknown) rate of an action is denoted by the top symbol T. The prefix operator can be used to model the behaviour of sequential activities.
Choice (+) : P+Q . The composed components represent a system that behaves as either P or Q . The first component to complete will change the internal state of the system and the other component will be discarded.
Cooperation ($\bowtie$) : (P $\bowtie_{\mathbf{L}}$ Q) . The cooperation operator is used to model the synchronisation behaviour of the system's components. The operator uses a cooperation set **L** to define the shared actions between the cooperating components P and Q . When the cooperation set is empty the activities of the two components will run in parallel $P \mid\mid Q$.
Hiding: P/L . The operator / is used to hide a set of local actions denoted by L and make them private to their component P . In PEPA language, the hidden (local) actions are denoted by the tau symbol τ.
Constant: A$\stackrel{\text{def}}{=}$P . Constant is used to associate a name like A with a component's behaviour like P.

PEPA eclipse plugin. PEPA plugin is a tool for editing, compiling, and deriving the state space of PEPA models. PEPA plugin can do static checking to detect errors and warnings in PEPA code and detect the deadlocks in PEPA models. Some of the warnings could be a result of unused processes and rates, and unused activities in cooperation sets. Moreover, PEPA plugin uses two well-known algorithms for performing stochastic simulation [8]: Gillespie's Stochastic Simulation Algorithm (SSA) and the Gibson- Bruck algorithm. More details about PEPA and PEPA eclipse plugin can be found in [12] and [26] respectively.

2.3 Related Research

Lamprecht et al. in [15] presented direct performance measurements of different implementations of different cryptographic protocols, which showed that measuring the performance of security protocols could be influenced by how the protocol was implemented. Thus, during the performance evaluation process, abstracting how the security protocol is implemented and the software/hardware configurations used to run the protocol will lead to a more generalised and formal performance evaluation of the security protocols .

In the past, there were some attempts to construct formal specifications either to verify the correctness of security protocols using formal specification languages such as communicating sequential processes (CSP), Event-B, and applied pi calculus, or to quantify and analyse the performance characteristics of security protocols using formal performance modelling paradigms such as queuing networks, petri-nets [28,29] and stochastic process algebra.

In proving security requirement correctness, Stathakidis, Schneider, and Heather, [21] modelled the four stages of Ximix Mix net protocol and also modelled an intruder using the CSP process algebra and FDR to verify the protocol's correctness. First, Stathakidis et al. modelled the mix net components with absence of dishonest participants and secondly modelled the Mix net protocol in

the presence of an adversary. Moreover, Culnane and Schneider in [5] modelled and analysed the behaviour of the bulletin board using Event-B formal specification language to verify the correctness of an ideal implementation of bulletin board and the correctness of the behaviour of the bulletin board in the presence of malicious participant. Furthermore, Kremer and Ryan [14] formalised three security properties of Fujioka, Okamoto and Ohta scheme [7] using the applied pi calculus language and they could prove that two of them were satisfied using ProVerif tool.

In formal evaluation of the performance of security protocols, Zhao and Thomas in [29] constructed a PEPA model for the exchange of secrete keys for n pairs of participants using a trusted third party. They used the protocol description of Stallings [27] to demonstrate how the KDC protocol behaves and used PEPA to study the scalability of the model. The authors in [29] derived the average utilisation of the KDC, and the average number of waiting requests in the key distribution centre. In [24] , Thomas and Zhao used the fluid flow analysis suggested by Hillston [13] to derive certain performance averages from the PEPA model of KDC that they modelled in [29]. They used the ordinary differential equations, ODEs, that represent the PEPA model of KDC to derive the number of waiting clients in the KDC system, and to derive the response time of the KDC.

Moreover, Zhao and Thomas in [28] modelled Zhou and Gollman non-repudiation protocol using PEPA formalism and applied two performance analysis techniques for deriving performance metrics for the modelled protocol. The two analysis approaches were the mean value analysis and the ordinary differential equations. The authors used mean value analysis MVA to calculate average response time and average number of derivatives, and used the fluid flow approximation based ODEs of the underlying PEPA models to drive more performance metrics such as average response time and queue length of waiting customers. Without generating the state space for the PEPA models of Zhou and Gollman non-repudiation protocol and with less computational complexity, the authors, in [25], calculated certain average metrics using MVA.

For electronic voting performance evaluation using PEPA, Thomas in [23] used PEAP to construct an electronic voting models for n voters with fault recovery for faulty voters. The performance models in [23] captured the voting scheme of Fujioka, Okamoto and Ohta [7] and two models constructed , one was to represent the basic behaviour of the protocol and the other to represent a simplified behaviour, and both models suffered from state-space explosion problem when modelling large number of voters. To model large number of voters , Thomas modified the basic model to make it represent a closed queuing network and called the last model queue-based model. The numerical results for analysing the three models showed that the number of voters in queue-based model scale up better that the other models in terms of number of transitions or states.

In [2] , Bradley and Gilmore used a stochastic simulation technique to convert a PEPA model of an e-voting scheme to a set of rate equations. Each rate equation represented an individual action of a component inside the PEPA model and by using these rate equations they could build a simulation description file that fitted the Dizzy simulation tool. Therefore, they could simulate and analyse the PEPA model for large number of voters. Manolopoulos et al. in [17] modelled the architecture's behaviour of the e-voting systems as open

Jackson networks of queues to evaluate the performance of the architecture. They also presented a simulation software that they used to simulate their theoretical model and derived performance metrics such e-voting server utilisation and average response time of the server. Finally, Ribarski and Antovski in [19] implemented the decrypt and re-encrypt mix-nets protocols in Java and evaluated the time it takes to pass private messages between sender and receiver through mix-nets using different key lengths of public key crypto-systems.

3 DRE-i PEPA Model and Analysis

Hao, Randell, and Clarke in [10] and Hao et al. in [11] presented DRE-i (Direct Recording electronic with integrity) e-voting scheme. The scheme is an end-to-end verifiable and self-enforcing cryptographic voting scheme based on the Direct Recording Electronic voting systems which replaces the tallying authority with a cryptographic homomorphic tallying algorithm. This scheme can be used in controlled or uncontrolled internet voting environment for large scale country-wide political elections or small-size elections like university students' unions elections.

3.1 DRE-i Electronic Voting Scheme

In this scheme, for each eligible voter n ballots will be generated inside a temperprove security module of the e-voting server and each ballot will have two encrypted candidates known as cryptograms. The voter using the voting client will prove his eligibility for voting to the voting server and will request a ballot. Upon receiving the ballot, the voter will choose one of the two cryptograms and send his selection to the server. The server will sign the received ballot and send the signed ballot to the voter to either accept it and cast it as his vote or reveal the signed ballot to verify that his selection reflects his intention. This cast-or-verify game that the voter plays with the e-voting server represents one part of individual verifiability that enables the voter to verify that his vote has been recorded as intended. When the voter chooses to cast, the voting client will send the voter's ballot to the voting server and in turn the voting server when it receives that ballot will send an acknowledgment of receiving the ballot to the voting client. The voting client at this stage will print a transcript for the voter that represents the signed encrypted ballot and this ends the vote casting process from the voter side. On the other hand, when the voter chooses to verify the integrity of the ballot that he is about to cast, the voting client will send a request to the e-voting server to decrypt the cryptogram that hides the chosen candidate and the server will reply with the decrypted cryptogram. The voting client will print the revealed cryptogram and will asks the e-voting server for another ballot so the voter can restart candidate selection process again. Figure.1 demonstrates the interactions between the voter, voting client and voting server to play the cast-verify game.

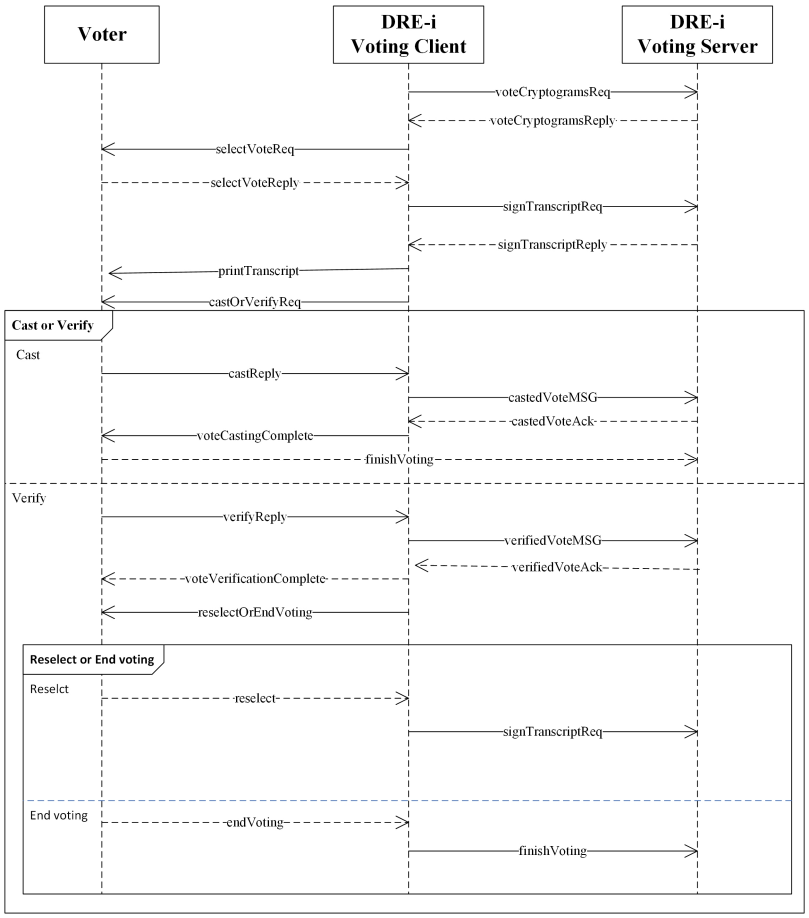

Figure 1: Cast-verify game of DRE-i e-voting scheme

The scheme utilises a public bulletin board to be used for publishing casted ballots and auditing information about the election process. This information will be used for tallying encrypted votes and for more verifiability actions.

3.2 PEPA Model

For the purpose of our research, we will be concerned about the behaviour of voter, voting client, and e-voting server that is associated with cast-verify game.

Voter:
$\mathbf{Voter_0}$ = (selectVoteReq,T).Voter_1
$\mathbf{Voter_1}$ = (selectVoteReply,$\text{r}_{\text{selectVoteReply}}$).Voter_2
$\mathbf{Voter_2}$ = (castOrVerifyReq,T).Voter_3
$\mathbf{Voter_3}$ = (castReply,$\text{r}_{\text{castReply}}$).$\text{Voter}_4$ +(verifyReply ,$\text{r}_{\text{verifyReply}}$).Voter_5
$\mathbf{Voter_4}$ = (voteCastingComplete,T).Voter_0
$\mathbf{Voter_5}$ = (voteVerificationComplete,T).Voter_6
$\mathbf{Voter_6}$ = (reselectOrEndVoting,T).Voter_7
$\mathbf{Voter_7}$ = (reselect,$\text{r}_{\text{reselect}}$).$\text{Voter}_8$ +(endVoting ,$\text{r}_{\text{endVoting}}$).$\text{Voter}_0$
$\mathbf{Voter_8}$ = (reselectVoteReq,T).Voter_1

Voting client:
$\mathbf{DRE\_Client_0}$ = (voteCryptogramsReq,$r_{\text{voteCryptogramsReq}}$).DRE_Client_1
$\mathbf{DRE\_Client_1}$ = (voteCryptogramsReply,T).DRE_Client_2
$\mathbf{DRE\_Client_2}$ = (selectVoteReq,$r_{\text{selectVoteReq}}$).DRE_Client_3
$\mathbf{DRE\_Client_3}$ = (selectVoteReply,T).DRE_Client_4
$\mathbf{DRE\_Client_4}$ = (signTranscriptReq,$r_{\text{signTranscriptReq}}$).DRE_Client_5
$\mathbf{DRE\_Client_5}$ = (signTranscriptReply,T).DRE_Client_6
$\mathbf{DRE\_Client_6}$ = (castOrVerifyReq,$r_{\text{castOrVerifyReq}}$).DRE_Client_7
$\mathbf{DRE\_Client_7}$ = (castReply,T).DRE_Client_8 + (verifyReply,T).DRE_Client_9
$\mathbf{DRE\_Client_8}$ = (castedVoteMSG,$r_{\text{castedVoteMSG}}$).DRE_Client_{10}
$\mathbf{DRE\_Client_{10}}$ = (castedVoteAck,$r_{\text{castedVoteAck}}$).DRE_Client_0
$\mathbf{DRE\_Client_9}$ = (verifiedVoteMSG,$r_{\text{verifiedVoteMSG}}$).DRE_Client_{11}
$\mathbf{DRE\_Client_{11}}$ = (verifiedVoteAck,T).DRE_Client_{12}
$\mathbf{DRE\_Client_{12}}$ = (reselectOrEndVoting,$r_{\text{reselectOrEndVoting}}$).DRE_Client_{13}
$\mathbf{DRE\_Client_{13}}$ = (reselect,T).DRE_Client_{14} + (endVoting,T).DRE_Client_0
$\mathbf{DRE\_Client_{14}}$ = (reselectVoteReq,$r_{\text{reselectVoteReq}}$).DRE_Client_3
Voting server:
$\mathbf{DRE\_SRV_0}$ = (voteCryptogramsReq,T).DRE_SRV_1
$\mathbf{DRE\_SRV_1}$ = (voteCryptogramsReply,$r_{\text{voteCryptogramsReply}}$).DRE_SRV_2
$\mathbf{DRE\_SRV_2}$ = (signTranscriptReq,T).DRE_SRV_3
$\mathbf{DRE\_SRV_3}$ = (signTranscriptReply,$r_{\text{signTranscriptReply}}$).DRE_SRV_4
$\mathbf{DRE\_SRV_4}$ = (castedVoteMSG,T).DRE_SRV_5 + (verifiedVoteMSG,T).DRE_SRV_6
$\mathbf{DRE\_SRV_5}$ = (castedVoteAck,$r_{\text{castedVoteAck}}$).DRE_SRV_0
$\mathbf{DRE\_SRV_6}$ = (verifiedVoteAck,$r_{\text{verifiedVoteAck}}$).$\text{DRE\_SRV}_7$
$\mathbf{DRE\_SRV_7}$ = (reselect,T).DRE_SRV_2 + (endVoting,T).DRE_SRV_0
System equation:

$$((\mathbf{DRE\_Client_0}[i] \bowtie_{L_1} \mathbf{DRE\_SRV_0}[j]) \bowtie_{L_2} \mathbf{Voter_0}[i])$$

where i is the number of voters in the system , j is the number of e-voting servers ,

$$\boldsymbol{L_1} = \left\{ \begin{array}{l} \text{voteCryptogramsReq, voteCryptogramsReply, signTranscript-} \\ \text{tReq,signTranscriptReply,castedVoteMSG, castedVoteAck, verified-} \\ \text{VoteMSG,verifiedVoteAck, reselect, endVoting} \end{array} \right\},$$

$$\boldsymbol{L_2} = \left\{ \begin{array}{l} \text{selectVoteReq, castOrVerifyReq, selectVoteReply, castReply, verifyRe-} \\ \text{ply,voteCastingComplete, voteVerificationComplete, reselectOrEndVoting,} \\ \text{reselect, endVoting, reselectVoteReq} \end{array} \right\}$$

3.3 Response Time Analysis

In the constructed PEPA model, we are interested in the performance overhead introduced by individual verifiability property, particularly the voter actions that he takes to verify that his vote was recorded correctly. The voters are interested in timely responses when they cast their votes and at the same time interested in verifying that their recorded votes reflect their intentions. Therefore, we will investigate the effect of the individual verification actions by voters on the average response time that voters expect when they cast their votes.

We constructed a PEPA model for the DRE-i e-voting scheme which was composed of three components: voter, voting client, and voting server. To study the average response time for casting a vote by a voter we will use littleś law:

$$\mathbf{L} = \lambda \mathbf{W} \tag{1}$$

where L is the population (the average number of jobs in the system), λ is the throughput, and W is the average time spent by the job inside the system.

Due to the state-space explosion of the PEPA model when modelling large number of components, the Eclipse PEPA plugin ran into the "Java heap space" error. However, using PEPA CTMC performance analysis approach we could calculate average response time for our PEPA model with small numbers of voters and DRE-i voting servers.

4 Conclusion

This paper showed the performance modelling and evaluation of the DRE-i e-voting scheme using PEPA. The model captured the e-voting scheme high-level interactions between voters, voting clients and voting servers with emphasis on the behaviour of the individual verifiability of the scheme. The analysis of response time of casting votes suffered from the state space explosion problem when we used the CTMC analysis and further performance analysis using more scalable performance evaluation techniques like fluid analysis and stochastic simulation will be investigated.

References

[1] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *2008 21st IEEE Computer Security Foundations Symposium*, pages 195–209. IEEE, 2008.

[2] Jeremy T Bradley and Stephen T Gilmore. Stochastic simulation methods applied to a secure electronic voting model. *Electronic Notes in Theoretical Computer Science*, 151(3):5–25, 2006.

[3] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[4] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[5] Chris Culnane and Steve Schneider. A peered bulletin board for robust use in verifiable voting systems. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 169–183. IEEE, 2014.

[6] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 10–18. Springer, 1984.

[7] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *International Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251. Springer, 1992.

[8] Stephen Gilmore and Jane Hillston. The pepa workbench: A tool to support a process algebra-based approach to performance modelling. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 353–368. Springer, 1994.

[9] Dimitris A Gritzalis. Principles and requirements for a secure e-voting system. *Computers & Security*, 21(6):539–556, 2002.

[10] Feng Hao, Dylan Clarke, and Carlton Shepherd. Verifiable classroom voting: Where cryptography meets pedagogy. In *Cambridge International Workshop on Security Protocols*, pages 245–254. Springer, 2013.

[11] Feng Hao, Matthew N Kreeger, Brian Randell, Dylan Clarke, Siamak F Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. In *2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14)*, 2014.

[12] Jane Hillston. *A compositional approach to performance modelling*, volume 12. Cambridge University Press, 2005.

[13] Jane Hillston. Fluid flow approximation of pepa models. In *Second International Conference on the Quantitative Evaluation of Systems (QEST'05)*, pages 33–42. IEEE, 2005.

[14] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *European Symposium on Programming*, pages 186–200. Springer, 2005.

[15] C Lamprecht, A Van Moorsel, P Tomlinson, and N Thomas. Investigating the efficiency of cryptographic algorithms in online transactions. *International Journal of Simulation: Systems, Science & Technology*, 7(2):63–75, 2006.

[16] Ülle Madise and Tarvi Martens. E-voting in estonia 2005. the first practice of country-wide binding internet voting in the world. *Electronic voting*, 86, 2006.

[17] Christos Manolopoulos, Dimitris Sofotassios, Paul Spirakis, Yannis C Stamatiou, and Yannis Stamatopoulos. Performance analysis of distributed, large-scale e-voting systems.

[18] Pascal Paillier and David Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 165–179. Springer, 1999.

[19] Pance Ribarski and Ljupcho Antovski. Mixnets: Implementation and performance evaluation of decryption and re-encryption types. *CIT. Journal of Computing and Information Technology*, 20(3):225–231, 2012.

[20] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[21] Efstathios Stathakidis, Steve Schneider, and James Heather. Robustness modelling and verification of a mix net protocol. In *International Conference on Research in Security Standardisation*, pages 131–150. Springer, 2014.

[22] Rajeev Syal. Eu referendum voter registration site crashes before deadline. urlhttp://www.theguardian.com/politics/2016/jun/07/eu-referendum-voter-registration-site-crashes-before-deadline, 6 2016. (Accessed on 08/27/2016).

[23] Nigel Thomas. Performability of a secure electronic voting algorithm. *Electronic Notes in Theoretical Computer Science*, 128(4):45–58, 2005.

[24] Nigel Thomas and Yishi Zhao. Fluid flow analysis of a model of a secure key distribution centre. In *Proceedings 24th Annual UK Performance Engineering Workshop, Imperial College London*, pages 44–57, 2008.

[25] Nigel Thomas and Yishi Zhao. Mean value analysis for a class of pepa models. In *European Performance Engineering Workshop*, pages 59–72. Springer, 2009.

[26] Mirco Tribastone, Adam Duguid, and Stephen Gilmore. The pepa eclipse plugin. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):28–33, 2009.

[27] Stallings William. Cryptography and network security: principles and practice. *Prentice-Hall, Inc*, pages 23–50, 1999.

[28] Y Zhao and N Thomas. *Efficient Analysis of PEPA model of Non-repudiation Protocols*. Citeseer, 2009.

[29] Yishi Zhao and Nigel Thomas. Approximate solution of a pepa model of a key distribution centre. In *SPEC International Performance Evaluation Workshop*, pages 44–57. Springer, 2008.